游荡在思考的迷宫中

甄景贤 (King-Yin Yan)

General.Intelligence@Gmail.com

Abstract. 介绍一个基於 **增强学习**和 **深度学习**的极简约的 cognitive architecture，它在数学上是一个 Hamiltonian 系统，而其 Lagrangian 对应於智能系统的「奖励」或「desire 的价值」。经典逻辑 AI 的技巧可以搬到这个 setting 之下，而连续时间化之后，可以用上微分几何的技巧。

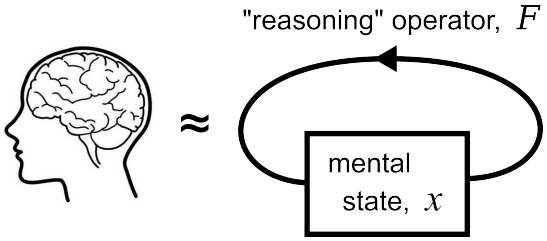
本文较少原创内容，主要介绍一些已知的理论，及提供一个新的观点，或许对 strong AI 的发展有帮助。

1 中心思想

标题中的**比喻**是指用增强学习的方法控制一隻自主的智能系统 (autonomous agent)，在「思维空间」中寻找最优路径：

(1)

关键是将「思考」看成是一个动态系统 (dynamical system)，它运行在思维状态 (mental states) 的空间中：



(2)

举例来说，一个思维状态可以是以下的一束命题：

- 我在我的房间内，正在写一篇 AGI-16 的论文。
- 我正在写一句句子的开头：「举例来说，....」
- 我将会写一个 NP (noun phrase)：「一个思维状态....」

思考的过程就是从一个思维状态 过渡 (transition) 到另一个思维状态。就算我现在说话，我的脑子也是靠思维状态记住我说话说到句子结构的哪部分，所以我才能组织句子的语法。

思维状态是一支向量 $\boldsymbol{x} \in X$，X 是所有可能的思维状态，思考算子 (reasoning operator) $\boldsymbol{F}$ 是一个 endomorphism 映射：$X \to X$。

在数学上这是一个标准的动态系统 (dynamical system)，它可以用以下方法定义：

$$\text{离散时间：} \quad \boldsymbol{x}_{t+1} = \boldsymbol{F}(\boldsymbol{x}_t) \tag{3}$$

$$\text{连续时间：} \quad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{4}$$

强化学习 (reinforcement learning)、动态规划 (dynamic programming)、控制论 (control theory) 的状态空间表述　三者其实是同义词；在人工智能里习惯叫 RL。

换句话说：我们将逻辑 AI 的整套器材搬到向量空间中去实现。这个做法，部分是受到 Google 的 PageRank [4] 和 Word2Vec [3] 算法的启发，因为它们都是在向量空间中运作，而且非常成功。

2 控制论

一个动态系统 (dynamical system) 可以用以下方法定义：

$$\text{离散时间：} \quad \boldsymbol{x}_{t+1} = \boldsymbol{F}(\boldsymbol{x}_t) \tag{5}$$

$$\text{连续时间：} \quad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) \tag{6}$$

其中 f 也可以随时间改变。如果 f 不依赖时间，则系统是 time-invariant （定常的），形式上如 (6) 那种微分方程叫作 autonomous （自主的）。

在我的智能系统理论里，我把 F 或 f 设定成 RNN (recurrent neural network)，即反馈式神经网络：

$$\text{离散时间：} \quad \boldsymbol{x}_{t+1} = \boxed{\text{RNN}}(\boldsymbol{x}_t) \tag{7}$$

$$\text{连续时间：} \quad \dot{\boldsymbol{x}} = \boxed{\text{RNN}}(\boldsymbol{x}) \tag{8}$$

这里 recurrent 指的是它不断重复作用在 $\boldsymbol{x}$ 之上，但实际上它是一个普通的前馈式 (feed-forward) 神经网络。注意：在抽象理论中，f 和 F 可以是任意函数，我把它们设计成 NN 只是众多可能的想法之一。之所以选用 NN，是因为它有 universal function approximator 的功能，而且是我们所知的最「聪明」的学习机器之一。

在我提出的智能系统里，$\dot{x}$ 是由學習機器給出的，換句話說，$\dot{x}$ 是思維狀態在梯度下降至最佳狀態時的方向導數。

一个（连续时间的）控制系统 (control system) 定义为：

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t), t) \tag{9}$$

其中 $\boldsymbol{u}(t)$ 是控制向量。控制论的目的就是找出最好的 $\boldsymbol{u}(t)$ 函数，令系统由初始状态 $\boldsymbol{x}_0$ 去到终点状态 $\boldsymbol{x}_\perp$。

注意：人工智能中的 **A* search**，是动态规划的一个特例。换句话说，用动态规划在某个空间中「漫游」，可以模拟到 best-first 搜寻的功能。

在这框架下，智能系统的运作可以分开成两方面：思考 和 学习。

思考即是根据已学得的知识（知识储存在 RNN 里），在思维空间中找寻 $\boldsymbol{x}$ 最优的轨迹，方法是用控制论计算 $\boldsymbol{u}^*$。$\boldsymbol{x}$ 的轨迹受 RNN 约束（系统只能依据「正确」的知识去思考），但思考时 RNN 是不变的。

学习就是学习神经网络 RNN 的 weights W。此时令 $u = 0$，即忽略控制论方面。

以上两者是两个独立的方面，但不排除它们可以在实际中同时进行。

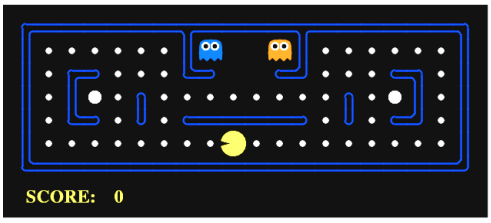
3 什么是强化学习？

Reinforcement learning 是机器学习里面的一个分支，特别善於控制一只能够在某个环境下自主行动 的个体 (autonomous agent)，透过和 环境 之间的互动，例如 sensory perception 和 rewards，而不断改进它的 行为。

听到强化学习，你脑里应该浮现一只甲由那样的小昆虫，那就是 autonomous agent 的形象：

(10)

对「环境」(environment) 这概念，你应该想到像以下这经典游戏的迷宫：



(11)

包括有追捕你的怪物、和吃了会加分的食物（这些代表负值和正值的 rewards）。当然，实际应用的「环境」和「奖励」可以是抽象的，这游戏是一个很具体的例子。

3.1 输入 / 输出

记住，reinforcement learning 的 输入 是：

- 状态 (States) = 环境，例如迷宫的每一格是一个 state
- 动作 (Actions) = 在每个状态下，有什么行动是容许的
- 奖励 (Rewards) = 进入每个状态时，能带来正面或负面的价值 (utility)

而输出就是：

- 方案 (Policy) = 在每个状态下，你会选择哪个行动？

於是这 4 个元素的 tuple $(S\ A\ R\ P)$ 就构成了一个强化学习的系统。在抽象代数中我们常常用这 tuple 的方法去定义系统或结构。

再详细一点的例子就是：

- states S = 迷宫中每一格的位置，可以用一对座标表示，例如 (1,3)
- actions A = 在迷宫中每一格，你可以行走的方向，例如：{ 上，下，左，右 }
- rewards R = 当前的状态 (current state) 之下，迷宫中的那格可能有食物 (+1) 、也可能有怪兽 (-100)
- policy P = 一个由状态 $\rightarrow$ 行动的函数，意即：这函数对给定的每一个状态，都会给出一个行动。

(S, A, R) 是使用者设定的，P 是算法自动计算出来的。

3.2 人与虫之间

第一个想到的问题是：为什么不用这个方法打造人工智能？但现时的强化学习算法，只对比较细小和简单的环境适用，对於大的复杂的世界，例如象棋的 10^{XXX} 状态空间，仍是 intractable 的。

关键就是，高等智慧生物会在脑中建立世界的模型 (world model) 或知识 (knowledge)，而强化学习只是关心简单的「状态－行动」配对。

强化学习的领导研究者 Richard Sutton 认为，只有这种学习法才考虑到自主个体、环境、奖励等因素，所以它是人工智能中最 top-level 的 architecture，而其他人工智能的子系统，例如 logic 或 pattern recognition，都应该在它的控制之下，我觉得颇合理。

(12)

所以要制造 strong AI，一个可能的方案就是结合强化学习和某种处理复杂 world model 的能力：

(13)

「你们已经由虫进化成人，但在你们之内大部份仍是虫。」

— 尼采, Thus spoke Zarathustra

「如果人类不相信他们有一天会变成神，他们就肯定会变成虫。」

— Henry Miller

3.3 強化學習的原理

《AI — a modern approach》这本书第 21 章有很好的简介。《AIMA》自然是经典，很多人说他们是读这本书而爱上 AI 的。这本书好处是，用文字很耐性地解释所有概念和原理，思路很清晰，使读者不致有杂乱无章的感觉。例如 21 章首先讲 passive reinforcement learning，意思是当 policy 是固定时，纯粹计算一下 agent 期望的价值（utility，即 rewards 的总和）会是多少。有了这基础后再比较不同 policies 的好坏。这种思路在数学中很常见：首先考虑简单到连白痴也可以解决的 case，然后逐步引入更多的复杂性。例如数学归纳法，由 $N = 1$ 的 case 推到 $N \to \infty$。

为免重复，我只解释到明白 Q learning 的最少知识。

3.4 Utility (价值，或效)

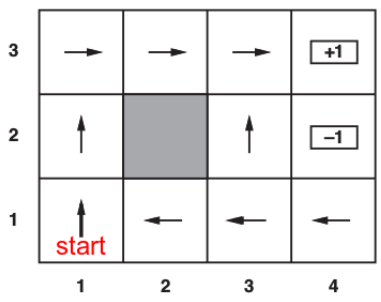
U 是一连串行动的 rewards 的总和。例如说，行一步棋的效用，不单是那步棋当前的利益，还包括走那步棋之后带来的后果。例如，当下贪吃一只卒，但 10 步后可能被将死。又或者，眼前有美味的食物，但有些人选择不吃，因为怕吃了会变肥。

一个 state 的效用 U 就是：假设方案固定，考虑到未来所有可能的 transitions，从这个 state 开始的平均期望的 total reward 是多少:

$$U(S_0) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t \, R(S_t) \,] \tag{14}$$

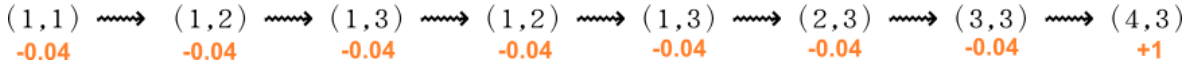
其中 $\mathbb{E}[\,]$ 代表期望值，γ 是 discount factor，例如 0.9 或什么。

实例：考虑这简单的迷宫：



那些箭咀表示的是众多可能方案中的其中一个。

根据这个方案，由 $(1,1)$ 开始的运行可能是这个结果：

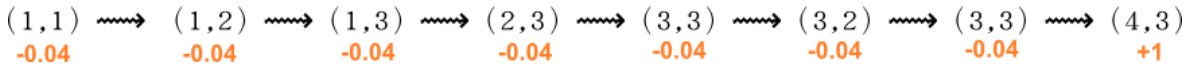


(15)

下面橙色的值是每个 state 的 reward。在这例子中，每个不是终点的格，也会扣 0.04 分。

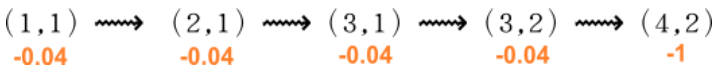
但从同一起点，同一方案，也可以出现不同结果，例如在 (1,3) 企图向右爬，但实际结果是向下跌一格；这些 state transitions 是由外在世界的机率决定的。（例如某人读了大学文凭，但遇上经济不景，他的薪水未必能达到行动的预期效果。）

同一方案的运行结果可以是：



(16)

或者：



(17)

3.5 Bellman condition

这是 dynamic programming（动态规划）的中心思想，又叫 Bellman optimality condition。

在人工智能里我们叫 reinforcement learning，但在控制论的术语里叫 dynamic programming，两者其实是一样的。Richard Bellman 在 1953 年提出这个方程，当时他在 RAND 公司工作，处理的是运筹学的问题。他也首先使用了"curse of dimensionality" 这个术语，形容动态规划的主要障碍。

考虑的问题是：要做一连串的 sequential decisions。

Bellman equation 说的是：「**如果从最佳选择的路径的末端截除一小部分，馀下的路径仍然是最佳路径。**」

换句话说，如果一系列的选择 A B C D E.... 是最优的，那么这系列除去开始的 A，那 B C D E.... 系列应用在后继的状态上也是最优的。

（例如，你从香港乘车到北京，选择了最便宜的路线，此路线经过 10 个车站，第二站是深圳：

$$\text{香港} \to \text{深圳} \to \ldots\ldots \to \text{北京}$$

但如果除去出发点香港站，那么由第二站深圳到最后的北京站：

$$\text{深圳} \to \ldots\ldots \to \text{北京}$$

这路线仍然是馀下 9 个站之间最便宜的。）

用数学表示：

$$U^*(S) = \max_a \{R(a) + U^*(S')\}$$

$$U^*(\text{全路径}) = \max_a \{R(\text{在当前状态下选取 a}) + U^*()\}$$

$*$ 表示 最优 (optimal)。这条看似简单的式子是动态规划的全部内容；它的意义是：我们想获得最佳效益的路径，所以将路径切短一些，於是问题化解成一个较小的问题；换句话说它是一个 recursive relation。

3.6 Delta rule

这只是一个简单的 trick，在机器学习中经常出现。假设我们有一个理想，我们要逐步调教当前的状态，令它慢慢趋近这个理想。方法是：

$$\text{当前状态} := \text{当前状态} + \alpha(\text{理想} - \text{当前状态})$$

其中 α 叫「学习速度 (learning rate)」。"Delta" (Δ) 指的是理想和现状之间的差异。

很明显，只要反覆执行上式，状态就会逐渐逼近理想值。

（Delta rule 的微分形式就是我们熟悉的「梯度下降」：$x \mathrel{+}= \eta \cdot \frac{dy}{dx}$）

将 delta rule 应用到 Bellman condition 上，去寻找最优路径，这就是 Temporal Difference (TD) learning。

3.7 Q value

Q 值只是 U 值的一个变种; U 是对每个 state 而言的，Q 把 U 值分拆成每个 state 中的每个 action 的份量。换句话说，Q 就是在 state S 做 action A 的 utility。

Q 和 U 之间的关系是：

$$U(S) = \max_A Q(A, S)$$

Q 的好处是什么？下面将会介绍 active learning，而 Q value 配合 TD learning，可以在 active learning 中也消除 P，达到 model-free 的效果。

上面的 update rule 只要用这个关系改写就行：

$$U(S) \mathrel{+}= \alpha(\ R(S) + \gamma \max_{A'} Q(A', S') - Q(A, S)\ )$$

4 控制论

在强化学习中，我们关注两个数量：

- $R(\boldsymbol{x}, a)$ = 在状态 $\boldsymbol{x}$ 做动作 a 所获得的奖励(reward)

- $U(\boldsymbol{x})$ = 状态 $\boldsymbol{x}$ 的效用(utility) 或 价值 (value)

简单来说，「价值」就是每个瞬时「奖励」对时间的积分：

$$\boxed{\text{价值 U}} = \int \boxed{\text{奖励 R}}\, dt \tag{18}$$

（价值有时用 V 表示，但为避免和势能 V 混淆故不用。）

用控制论的术语，通常定义 cost functional：

$$J = \int L dt + \Phi(\boldsymbol{x}_\perp) \tag{19}$$

其中 L 是 "running cost"，即行走每一步的「价钱」；Φ 是 terminal cost，即到达终点 $\boldsymbol{x}_\perp$ 时，那位置的价值。

在分析力学里 L 又叫 Lagrangian，而 L 对时间的积分叫「作用量」：

$$\boxed{\text{作用量 (Action)}\quad \text{S}} = \int L dt \tag{20}$$

Hamilton 的最小作用量原理 (principle of least action) 说，在自然界的运动轨迹里，S 的值总是取稳定值 (stationary value)，即比起邻近的轨迹它的 S 值最小。

所以有这些对应：

强化学习	最优控制	分析力学
效用/价值 U	价钱 J	作用量 S
即时奖励 R	running cost	Lagrangian L
action a	control u	（外力?）

用比较浅显的例子：和美女做爱能带来即时的快感 (= 奖励 R)，但如果强奸的话会坐牢，之后很长时间很苦闷，所以这个做法的长远价值 U 比其他做法较低，正常人不会选择它。

有趣的是，奖励 R 对应於力学上的 Lagrangian，其物理学单位是「能量」；换句话说，「快感」或「开心」似乎可以用「能量」的单位来量度，这和通俗心理学里常说的「正能量」不谋而合。而，长远的价值，是以 [能量 × 时间] 的单位来量度。

一个智能系统，它有「智慧」的条件，就是每时每刻都不断追求「开心能量」或奖励 R 的最大值，但它必需权衡轻重，有计划地找到长远的效用 U 的最大值。

5 Episodic memory

设计了这个 minimalist architecture 之后，发现比较起人脑有个严重缺陷，就是没有「事件记忆」（换句话说，只能留意当下发生的事件，但不能记住一段故事）。

这牵涉到甚么是「记忆」的问题。在 minimal architecture 里，$\boldsymbol{F}$ 代表 “static knowledge”，亦即（相对地）永恒不变的规律，而 $\boldsymbol{x}$ 代表当下的状态，亦即 “dynamic knowledge”。

Episodic memory 介乎「动态」与「静态」之间，我估计和信息处理理论中的 z-transform 或许有关。Episodic memory 的问题仍有待研究，但没有 episodic memory 也可以制造一种颇为有用的智能系统了。

6 经典逻辑 AI

Strong AI 的问题在理论上已经被数理逻辑完整地描述了，馀下的问题是学习算法，因为在逻辑 AI 的架构下，学习算法很慢（复杂性很高），这就是我们要解决的。

我研究 logic-based AI 很多年，因此我的思路喜欢将新问题还原到逻辑 AI 那边去理解，但实际上我提倡的解决办法不是靠经典逻辑，甚至不是 symbolic 的。但在这篇文章我还是会经常跳回到逻辑 AI 去方便理解。

用数理逻辑模拟人的思想是可行的，例如有 deduction, abduction, induction 等这些模式，详细可见《Computational logic and human thinking》by Robert Kowalski, 2011. 这些方面不影响本文的阅读。值得一提的是，作者 Kowalski 是 logic programming，特别是 Prolog，的理论奠基人之一。

在经典逻辑 AI 中，「思考」是透过一些类似以下的步骤：

$$\text{前提} \vdash \text{结论} \tag{21}$$

$$\boxed{\text{今天早上下雨}} \vdash \boxed{\text{草地是湿的}} \tag{22}$$

亦即由一些命题(propositions) 推导到另一些命题。

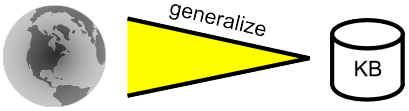
推导必须依靠一些逻辑的法则命题 (rule propositions)，所谓「法则」是指命题里面带有 x 这样的变量(variables)：

$$\boxed{\text{地方 x 下雨}} \wedge \boxed{\text{x 是露天的}} \vdash \boxed{\text{地方 x 是湿的}} \tag{23}$$

这些法则好比「逻辑引擎」的燃料，没有燃料引擎是不能推动的。

注意：命题里面的 x，好比是有「洞」的命题，它可以透过 substitution 代入一些实物 (objects)，而变成完整的命题。这种「句子内部」(sub-propositional) 的结构可以用 predicate logic （谓词逻辑）表达，但暂时不需要理会这些细节。

Logic-based AI 可以看成是将世界的「模型」压缩成一个「知识库」(knowledge-base, KB)，里面装著大量逻辑式子：



(24)

世界模型是由大量的逻辑式子经过组合而生成的，有点像向量空间是由其「基底」生成；但这生成过程在逻辑中特别复杂，所以符号逻辑具有很高的压缩比，但要学习一套逻辑知识库，则相应地也有极高的复杂度。

Acknowledgements

In a forum discussion with Ben Goertzel dated 25 June 2014 on the AGI mailing-list: (artificial-general-intelligence @googlegroups.com), YKY asked: Why bother with neural networks, which typically require many neurons to encode data, when logic-based AI can represent a proposition with just a few symbols? Ben's insight is that neural networks are capable of learning its own representations, and their learning algorithms are relatively fast. We have been working on "neo-classical" logic-based AI for a long time, and begin to realize that inductive learning in logic (based on combinatorial search in a symbolic space) is perhaps *the bottleneck* in the entire logic-based paradigm. So we try to look for alternatives that might enable learning to be faster, though we would still emphasize that logic-based AI remains a viable approach to AGI.

References

1. Itamar Arel. *Deep reinforcement learning as Foundations for Artificial Intelligence*, chapter 6, pages 89–102. Atlantis Press, 2012.
2. Lo. Dynamical system identification by recurrent multilayer perceptrons. *Proceedings of the 1993 World Congress on Neural Networks*, 1993.
3. Mikolov, Sutskever, Chen, Corrado, and Dean. Efficient estimation of word representations in vector space. *Proceedings of workshop at ICLR*, 2013.
4. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.
5. Siegelmann and Sontag. Turing computability with neural nets. *Applied Mathematics Letters, vol 4, p77-80*, 1991.